Package ‘cape’

February 19, 2015

Type Package
Title Combined analysis of pleiotropy and epistasis
Version 1.3
Date 2013-08-02
Author Anna L. Tyler, Wei Lu, Justin J. Hendrick, Vivek M. Philip, and Greg W. Carter
Maintainer Anna L. Tyler <Anna.Tyler@jax.org>
Description This package combines complementary information across multiple related phenotypes to infer directed epistatic interactions between genetic markers. The analysis in this package can be applied to a variety of engineered and natural populations.
Depends R (>= 3.1.1)
Imports corpcor, evd, qpcR, Matrix, igraph, fdrtool, shape
License GPL-3
NeedsCompilation no
Repository CRAN
Date/Publication 2014-09-26 18:50:25

R topics documented:

  cape-package ........................................... 2
calc.p .................................................. 3
create.covar ........................................... 4
delete.pheno .......................................... 5
direct.influence ....................................... 5
error.prop .............................................. 7
get.covar .............................................. 8
get.eigentraits ......................................... 8
get.network ............................................ 9
norm.pheno ............................................. 11
obesity.cross .......................................... 12
pairscan ................................................ 12
plotCollapsedVarInf .................................. 14
plotNetwork .......................................... 15
Description

This package infers predictive networks between genetic variants and between genetic variants and phenotypes. It uses complementary information of pleiotropic gene variants across different phenotypes to resolve models of epistatic interactions between genetic variants. To do this, cape reparameterizes main effect and interaction coefficients from a pairwise variant regressions into directed influence parameters. These parameters describe how gene variants influence each other, in terms of suppression and enhancement, as well as how gene variants influence phenotypes.

Details

Package: cape
Type: Package
Version: 1.3
Date: 2013-08-12
License: GPL-3

The cape analysis begins by reading in a genetic data set with \texttt{read.population}. The data are converted into a data object, to which results are added throughout the analysis. This data object is an argument in most functions, and is referred to as \texttt{data.obj}. Because this package uses pleiotropy to resolve models of epistasis, the phenotypes used should have common underlying molecular players, but not be perfectly correlated. In general phenotypes should be correlated with a Pearson \textit{r} between 0.4 and 0.8. The phenotypes of interest are then decomposed into eigentraits using \texttt{get.eigentraits}. Any number of phenotypes can be decomposed, but cape requires between
two and 12 eigentraits for the analysis. The phenotype decomposition into eigentraits maximizes the complementary information between the phenotypes. Before investigating epistatic interactions through a pair-wise scan of the genetic variants, a single-variant scan (singlescan) is run. This allows for thresholding of markers for the pair scan if the cross is prohibitively large to test all pairs of variants. The single-variant scan also allows selection of variants with very large main effects to be used as covariates in the pair scan. The pair scan (pairscan) performs a regression on each variant pair. Finally, the coefficients from the pairwise scan are reparameterized to yield directional influences between variants and from variants to phenotypes. The final result is an asymmetric adjacency matrix describing these variant influences. The p values of these influences are corrected for multiple tests.

Author(s)

Anna L. Tyler, Wei Lu, Justin J. Hendrick, Vivek M. Philip, and Gregory W. Carter Maintainer: Anna L. Tyler <anna.tyler@jax.org>

References


Examples

```r
data(obesity.cross)
str(obesity.cross)
```

---

**calc.p**

*Calculate P Values for Interactions Based on Permutations*

**Description**

This function uses the permutation results to calculate empirical p values for the variant-to-variant influences calculated by `error.prop`. It can also optionally adjust these p values using Holm’s step-down procedure, false discovery rate (fdr), or local false discovery rate (lfdr).

**Usage**

```r
calc.p(data.obj, pval.correction = c("holm", "fdr", "lfdr", "none"))
```

**Arguments**

- `data.obj` The object in which all results are stored. See `read.population`.
- `pval.correction` One of "holm", "fdr", "lfdr" or "none", indicating whether the p value correction method used should be the Holm step-down procedure, false discovery rate, local false discovery, or no correction rate respectively.
The data object is returned with a new list with two elements. The elements correspond to the two directions of influence: marker1 to marker2 and marker2 to marker1. Each element contains a table with the source and target variants, the empirical p values, and the adjusted p values, along with the effect size, standard error and t statistic for each interaction.

References


create.covar

Designate a phenotype or treatment as a covariate

Description

In cape, covariates are coded as genomic loci. To use a phenotype (e.g. male/female) or experimental treatment (e.g. case/control) as a covariate, this factor must be moved to the genotype matrix. create.covar removes the specified phenotype from the phenotype matrix and places it in the genotype matrix. It is assigned to chromosome 0.

Usage

create.covar(data.obj, pheno.which)

Arguments

data.obj The object in which all results are stored. See read.population.
pheno.which Either a column number or a character string indicating which phenotype should be transferred to the genotype matrix. If a character string is used, it must exactly identify the phenotype column header of interest.

Value

This function returns the data object with the designated phenotype removed from the phenotype matrix and added to the genotype matrix

Examples

data(obesity.cross)
str(obesity.cross)
obesity.cross <- create.covar(obesity.cross, pheno.which = "mom")
str(obesity.cross)
**delete.pheno**  
*Remove phenotypes from the phenotype matrix*

**Description**

This function deletes an unwanted phenotype or phenotypes from the phenotype matrix.

**Usage**

```r
delete.pheno(data.obj, phenotypes)
```

**Arguments**

- `data.obj`: The object in which all results are stored. See `read.population`.
- `phenotypes`: A vector of either column numbers or column names designating which phenotype or phenotypes should be deleted.

**Value**

This function returns the data object with the specified phenotypes removed.

**Examples**

```r
data(obesity.cross)
str(obesity.cross)
obesity.cross <- delete.pheno(obesity.cross, "insulin")
str(obesity.cross)
```

**direct.influence**  
*Calculate the significance of direct influences of variant pairs on phenotypes*

**Description**

This function recasts the variant-to-eigentrait effects in terms of variant-to-phenotype effects. It multiplies the $\beta$-coefficient matrices of each variant (i) and each phenotype (j) ($\beta_i^j$) by the singular value matrices ($V \cdot W^T$) obtained from the singular value decomposition performed in `get.eigentraits`.  

$\beta_i^j = V \cdot W^T$. It also uses the permutation data from the pairwise scan (`pairscan`) to calculate an empirical p value for the influence of each marker pair on each phenotype. The empirical p values are then adjusted for multiple testing using Holm’s step-down procedure.

**Usage**

```r
direct.influence(data.obj, transform.to.phenspace = TRUE, pval.correction = c("holm", "fdr", "lfdr"), verbose = FALSE, save.permutations = FALSE)
```
**direct.influence**

**Arguments**

- **data.obj**  The object in which all results are stored. See `read.population`.
- **transform.to.phenospace**  A logical value. If TRUE, the influence of each marker on each eigentrait is transformed to the influence of each marker on each of the original phenotypes. If FALSE, no transformation is made. If the pair scan was done on eigentraits, the influence of each marker on each eigentrait is calculated. If the pair scan was done on raw phenotypes, the influence of each marker on each phenotype is calculated. The default behavior is to transform variant influences on eigentraits to variant influences on phenotypes.
- **pval.correction**  One of "holm", "fdr", or "lfdr", indicating whether the p value correction method used should be the Holm step-down procedure, false discovery rate or local false discovery rate respectively.
- **verbose**  A logical value. If TRUE, the progress of the function is printed to the screen. If FALSE, the default, nothing is printed.
- **save.permutations**  A logical value indicating whether the data from permutations should be saved. Saving the permutations requires more memory but can be helpful in diagnostics. If `save.permutations` is TRUE all permutation data are saved in an object called "permutation.data.RData".

**Value**

This function adds a total of six objects to the data object. First, a flag (`transform.to.phenospace`) is added to the data object to indicate whether variant influences were transformed to phenotype space.

The results from the pairwise scan and the permutations of the pairwise scan are converted to phenospace if specified. These actions each add one object each to the data object (`var.to.pheno.influence` and `var.to.pheno.influence.perm`). Each element is itself a list of matrices corresponding to the original phenotypes. Each matrix contains one row per marker pair (or permutation of a marker pair) and contains the influence coefficient and standard error of the influence coefficient for each pair.

After the coefficients have been transformed to phenotype space, each marker is considered individually and its influence on each phenotype across all marker pair contexts is tabulated. This is done for both the pairwise scan and the permutations of the pairwise scans and adds two new objects (`var.to.pheno.test.stat` and `var.to.pheno.test.stat.perm`) to the data object. Each object is a list containing one element for each of the original phenotypes. Each element contains a table in which all instances of each marker are listed along with that marker's direct phenotypic influence, the standard error of the influence, and the t statistic ($\beta/\sigma$) of the influence.

Finally, because each marker can only have one influence on each phenotype, the influences from each marker pair context are filtered to report only the maximum influence of each marker on each phenotype across all marker pair contexts. This process adds an object to the data object called `max.var.to.pheno.influence`. This object is a list containing one element per phenotype. It tabulates the maximum influence of each marker on each phenotype, as well as the empirical and Holm’s corrected p values associated with each influence.
error.prop

References


error.prop

Estimate Errors of Regression Coefficients

Description

This function uses error propagation formulas for quantities computed from regression coefficients to estimate the error for all regression coefficients.

Usage

error.prop(data.obj, perm = FALSE, verbose = FALSE)

Arguments

data.obj The object in which all results are stored. See read.population.
perm A logical value to indicate whether error propagation should be performed on the test statistics (FALSE) or the permuted test statistics (TRUE).
verbose A logical value to indicate whether the progress of the function should be printed to the screen.

Value

This function returns the data object with a new list element: var.to.var.influences if perm is set to FALSE and var.to.var.influences.perm if perm is set to TRUE. These tables include the errors calculated for the marker1 to marker2 influences as well as the marker2 to marker1 influences. These results are used by calc.p to calculate empirical p values.

References


get.covar

Use a threshold to automatically select covariates for the pairscan

Description
This function uses a covariate threshold to calculate which markers will be used as covariates in the pair scan.

Usage
get.covar(data.obj, covar.thresh = NULL)

Arguments
- data.obj: The object in which all results are stored. See read.population.
- covar.thresh: A numerical value indicating the standardized effect size ($\beta/\sigma$) above which a marker is considered to have a strong enough effect to be used as a covariate.

Value
This function operates on the element in data.obj called covar.flags. This element is a table with one row for each marker and one column for each trait being analyzed. The entry for each marker contains a 1 if it is to be used as a covariate and a 0 otherwise.

References

See Also
set.covar, singlescan, select.markers.for.pairscan, pairscan

get.eigentraits
Calculate eigentraits from phenotype matrix

Description
This function performs the singular value decomposition (SVD) on the phenotype matrix after first removing individuals with missing data. The eigentraits are the left singular vectors of the decomposition. This function optionally mean centers and normalizes the phenotype matrix before performing the SVD.

Usage
get.eigentraits(data.obj, scale.pheno = TRUE, normalize.pheno = TRUE)
**get.network**

**Arguments**

- `data.obj` The object in which all results are stored. See `read.population`.
- `scale.pheno` A logical value specifying whether the phenotypes should be mean centered before the SVD is performed. The default, and recommended, value is TRUE.
- `normalize.pheno` A logical value specifying whether the phenotypes should be quantile normalized before the SVD is performed.

**Value**

This function adds three new elements to the data.obj list.

- `ET` The left singular vectors from the SVD. These are the eigentraits.
- `singular.values` The singular values from the SVD. These are used later internally to convert variant effects from eigentrait space to phenotype space.
- `right.singular.vectors` The right singular vectors from the SVD. These are used later internally to convert variant effects from eigentrait space to phenotype space.

**Note**

There must be more individuals than phenotypes to perform this calculation. An error results if there are more phenotypes than individuals.

**References**


**See Also**

`norm.pheno`, `plotSVD`

---

**get.network**  
*Convert the final results to a form plotted by plotNetwork and plotCollapsedVarInf*

**Description**

This function converts the significant epistatic interactions to a form that can be plotted as a network. This conversion also optionally condenses the network based on linkage between markers. The degree to which the network is condensed is determined by the argument `r2.thresh`. This value sets the correlation at which two markers are considered linked.
Usage

get.network(data.obj, p.or.q = 0.05, min.std.effect = 0,
collapse.linked.markers = TRUE, r.thresh = 0.5,
verbose = FALSE, plot.linkage.blocks = FALSE)

Arguments

data.obj The object in which all results are stored. See read.population.
p.or.q A numerical threshold indicating the maximum adjusted p value considered significant. If an fdr method has been used to correct for multiple testing, this value specifies the maximum q value considered significant.
min.std.effect A numerical threshold indicating the absolute value of the minimum standard effect size to be shown in the plot. The default value of 0 performs no thresholding.
collapse.linked.markers A logical value. If TRUE markers are combined into linkage blocks based on correlation. If FALSE, each marker is treated as an independent observation.
r.thresh If collapse.linked.markers is TRUE and linkage.method is set to "genotype", this numerical value indicates the Pearson r value at which two markers are considered linked. If two markers are correlated according to the threshold, they are collapsed into a single marker.
verbose A logical value indicating whether the function progress should be printed to the screen.
plot.linkage.blocks A logical value indicating whether the chromosomes should be plotted with their linkage blocks delineated. The type of plot produced differs depending on which choice is specified by linkage.method.

Details

This function calls defineds linkage blocks based on the correlation between adjacent markers. The first marker on the chromosome is the first marker of the initial linkage block. The algorithm steps through markers individually comparing each to the first marker in the block. When a marker has a correlation lower than the specified threshold when compared to the initial marker in the block, a boundary is drawn, and the marker becomes the first marker of the next block. This process is repeated until all markers on the chromosome are assigned to a block.

See Also

plotNetwork, plotCollapsedVarInf
norm.pheno

---

**Normalize and mean center phenotypes**

**Description**

This function performs quantile normalization on phenotypes and optionally mean centers them.

**Usage**

```
norm.pheno(data.obj, mean.center = TRUE)
```

**Arguments**

- `data.obj`: The object in which all results are stored. See `read.population`.
- `mean.center`: A logical value. If TRUE the phenotypes are mean centered in addition to being normalized. if FALSE, the phenotypes are not mean centered.

**Details**

In quantile normalization the values of the phenotype are sorted and replaced with a corresponding value drawn from a normal distribution with the same standard deviation and mean as the original distribution. Mean centering subtracts the mean phenotype value from each phenotype value yielding a distribution centered around 0.

**Value**

This function returns `data.obj` with the normalized phenotypes in place of the original phenotypes.

**Note**

Both normalization and mean centering are highly recommended before obtaining the eigentraits with singular value decomposition (SVD) (see `get.eigentraits`). This normalization procedure can also be performed by `get.eigentraits`.

**See Also**

- `get.eigentraits`

**Examples**

```r
t # T # be sure to move over any covariates before normalizing the phenotypes
## obesity.cross <- create.covar(obesity.cross, "mom")
# obesity.cross <- norm.pheno(obesity.cross)
hist(obesity.cross$pheno,"glucose", main = "Histogram of Normalized Glucose",

xlab = "Normalized Glucose (mg/dL)"
```
Mouse cross data from Reifsnyder et al. (2000)

Description

Data from a cross between non-obese, non-diabetic (NON) mice, and diabetes-prone New Zealand obese (NZO) mice. The experiment is described in Reifsnyder et al. (2000). The data object is a list. The first element is a matrix of phenotype data. It includes insulin levels (ng/mL), plasma glucose levels (mg/dL), total body weight (g), all measured at age 24 weeks. The second element is the genotype matrix containing the genotype of each mouse at each of 85 markers across the genome. The final elements are the chromosome vector, which indicates which chromosome on which each marker is found, and a vector of chromosomal positions indicating the location of each marker on the chromosome in centimorgans (cM).

Usage

data(obesity.cross)

Format

The format is: List of 4 $ pheno : num [1:204, 1:3] 63.3 31 43.3 33.3 35.3 15.9 23.6 NA 16 22 ...
..- attr(*, dimnames)=List of 2 ..$ : NULL ..$ : chr [1:3] body_weight sex $ geno : num [1:204, 1:85] 0.5 0.5 0.5 0.5 0.5 0.5 0.5 NA 0 0 ...
..- attr(*, dimnames)=List of 2 ..$ : chr [1:85] D1Mit296 D1Mit211 D1Mit411 D1Mit123 ...
$ chromosome : chr [1:85] 1 1 1 1 ...
$ marker.location: num [1:85] 2.08 10.59 12.62 17.67 22.88 ...

Source


pairscan

Perform regressions for all pairs of markers and all phenotypes.

Description

This function performs the pairwise regression on all selected marker pairs. The phenotypes used can be either eigentraits or raw phenotypes. Permutation testing is also performed.

Usage

pairscan(data.obj, scan.what = c("eigentraits", "raw.traits"), n.perm = NULL, min.per.genotype = NULL, max.pair.cor = NULL, verbose = FALSE, num.pairs.limit = 1e4, num.perm.limit = 1e7)
pairscan

Arguments

data.obj
The object in which all results are stored. See `read.population`.

scan.what
A character string uniquely identifying whether eigentraits or raw traits should be scanned.

n.perm
The number of permutations.

min.per.genotype
The minimum number of individuals allowable per genotype. If for a given marker pair, one of the genotypes is underrepresented, the marker pair is not tested. If this value is NULL, `max.pair.cor` must have a numeric value.

max.pair.cor
A numeric value between 0 and 1 indicating the maximum Pearson correlation that two markers are allowed. If the correlation between a pair of markers exceeds this threshold, the pair is not tested. If this value is set to NULL, `min.per.genotype` must have a numeric value.

verbose
A logical value indicating whether the progress of the scan should be printed to the screen.

num.pairs.limit
A number indicating the maximum number of pairs to scan. If the number of pairs exceeds this threshold, the function asks for confirmation before proceeding with the pairwise scan.

num.perm.limit
A number indicating the maximum number of total permutations that will be performed. If the number of total permutations exceeds this threshold, the function asks for confirmation before proceeding with the pairwise scan.

Details

Not all marker pairs are necessarily tested. Before markers are tested for interaction, they are checked for several conditions. Pairs are discarded if (1) at least one of the markers is on the X chromosome, or (2) there are fewer than `min.per.genotype` individuals in any of the genotype combinations.

Value

This function adds an element to the data object reporting the results of the pairwise scan:

`pairscan.results`
The results of the pairwise scan on the provided phenotype and genotypes.

If permutations have been performed (n.perm > 0), an additional element is added to the object reporting the results of the permutation tests:

`pairscan.perm`
The results of the permutations of the pairwise scan on the provided phenotype and genotypes.

Each of these results elements is itself a list of 3 elements:

`pairscan.effects`
A table of effects of each marker pair. The columns list the effects in the following order: marker1, marker2, the variance of marker1, the covariance of marker1
and marker2, the variance of marker2, the covariance of marker1 and the interaction effect, the covariance between marker2 and their interaction effect, and the variance of the interaction.

**pairscan.se** A table of the standard errors from the test on each marker pair. The columns are identical to those described for pairscan.effects

**model.covariance**
This is a table in which each row is the linearized matrix of the variance-covariance matrix of each pairwise regression.

The results element for the permutation tests has the same structure as for the pairwise scan except that each row represents the results of one permutation.

**References**


**See Also**

`select.markers.for.pairscan`, `plotPairscan`

**Examples**

```r
## Not run:
obesity.cross <- pairscan(obesity.cross, scan.what = "eigentraits", n.perm = 10,
min.per.genotype = 6, use.pairs.threshold = TRUE, verbose = TRUE)
## End(Not run)
```

---

**plotCollapsedVarInf**  *Plot variant-to-variant influences*

**Description**

This function plots the final network as does `plotVariantInfluences`, but it plots the network condensed by linkage blocks as performed by `get.network`. This function can only be run after running `get.network`.

**Usage**

```r
plotCollapsedVarInf(data.obj, expand.labels = FALSE,
all.markers = FALSE, scale.effects = c("log10", "sqrt", "none"))
```
**plotNetwork**

**Arguments**

- **data.obj**: The object in which all results are stored. See `read.population`.
- **expand.labels**: A logical value. If FALSE, the markers are labeled as linkage blocks ("block1", "block2" and so forth). If TRUE, the block labels are expanded to show all-marker names included in each block.
- **all.markers**: A logical value. If TRUE all markers are plotted. If FALSE only markers tested in the pair scan are plotted.
- **scale.effects**: A string indicating a scaling function by which the effects should be scaled. This is useful in increasing contrast between effects with large variance.

**See Also**

`pairscan`

---

**plotNetwork**

*Plot the final epistatic network*

**Description**

This function plots the final results with a using a different layout than `plotVariantInfluences`. Instead of the adjacency matrix, this function plots interactions between markers with arrows indicating the direction of influence. The function `get.network` must be run before plotting the network.

**Usage**

```r
plotNetwork(data.obj, collapsed.net = TRUE, trait = NULL,
phenotype.labels = NULL, main.lwd = 4, inter.lwd = 3,
label.cex = 1.5, percent.bend = 15, chr.gap = 1,
label.gap = 5, positive.col = "#af8dc3",
negative.col = "#7fbF7b")
```

**Arguments**

- **data.obj**: The object in which all results are stored. See `read.population`.
- **collapsed.net**: A logical value. If TRUE, a network condensed by linkage (see `get.network`) is plotted. If FALSE, the full network is plotted.
- **trait**: A character vector indicating which traits should be plotted. If NULL, all traits are plotted.
- **phenotype.labels**: A vector of strings listing alternate names for the phenotypes.
- **main.lwd**: A numeric value specifying the line width of the main effects bars surrounding the chromosome bars.
- **inter.lwd**: A numeric value specifying the line width of the interaction arrows drawn inside the chromosome circle.
plotPairscan

plot the results from pairscan

Description

This function plots the results from the pairwise regressions. The matrices are plotted with results coded on a yellow to blue scale. Results from all phenotypes are plotted on the same scale.

Usage

plotPairscan(data.obj, phenotype = NULL, standardized = TRUE,
             show.marker.labels = FALSE, show.chr = TRUE,
             label.chr = TRUE, pdf.label = "Pairscan.Regression.pdf",
             verbose = FALSE)

Arguments

data.obj The object in which all results are stored. See read.population.
phenotype A character vector indicating which phenotypes should be plotted. All phenotypes are plotted if phenotype is set to NULL.
`plotSinglescan` 17

standardized A logical value. If FALSE, the plotted values are the beta coefficients of the interaction in the pairwise model. If TRUE, the t statistics ($\beta/\sigma$) are plotted.

show.marker.labels A logical value indicating whether marker labels should appear on the axes.

show.chr A logical value. If TRUE, chromosomes are indicated by alternating gray and white blocks along plot axes.

label.chr A logical value. If TRUE and if show.chr is TRUE, chromosome numbers are printed inside each gray and white block.

pdf.label A character string indicating the name of the file that the plots should be printed to. When multiple phenotypes are plotted, all phenotypes are plotted on the same scale and each is plotted in a different page of the pdf.

verbose A logical value indicating whether the progress of the plotting function should be reported. Large data sets can take a long time to process and plot.

Value
No values are returned.

See Also
`pairsscan`

**Description**
This function plots the results obtained from the single-marker regression performed by singlescan. The effects ($\beta$) of each regression on each phenotype or eigentrait are plotted as a vertical line. Chromosomes and traits for plotting can be specified.

**Usage**

```r
plotSinglescan(data.obj, chr = NULL, traits = NULL,
               show.alpha.values = NULL, standardized = TRUE,
               show.marker.labels = FALSE, mark.covar = TRUE, mark.chr = TRUE,
               plot.type = "h", overlay = FALSE, trait.colors = NULL,
               show.rejected.markers = FALSE, show.selected.markers = FALSE)
```

**Arguments**

data.obj The object in which all results are stored. See `read.population`.

chr An optional vector indicating which chromosomes should be plotted. If NULL, the default, all chromosomes are plotted.

traits An optional vector indicating which traits should be plotted. If NULL, the default, all traits are plotted.
show.alpha.values
A numeric vector indicating which alpha values specified in singlescan should be plotted. If NULL, all alpha values calculated in singlescan are plotted.

standardized
A logical value. If TRUE, the absolute value of the regression t statistics ($\beta/\sigma$) are plotted. If FALSE, the raw regression coefficients ($\beta$) are plotted.

show.marker.labels
A logical value indicating whether marker names should be printed along the plot axes.

mark.covar
A logical value. If TRUE, the covarates for the pair scan are marked in red. If FALSE, all markers are plotted in black.

mark.chr
A logical value. If TRUE, alternating chromosomes are shaded in gray to aid in visualizing chromosome boundaries.

plot.type
A character indicating whether the marker effects should be plotted as vertical lines ("h"), points ("p"), lines ("l"), or both lines and points ("b").

overlay
A logical value indicating whether plots of scans should be overlayed on top of each other or plotted in separate panels.

trait.colors
If overlay is TRUE, this argument specifies different colors for the overlayed scans. There are four default colors (black, blue, purple, darkgreen). Alternate or additional colors can be specified manually.

show.rejected.markers
A logical value. If select.markers.for.pairscan has been run, setting this value to TRUE places indicators above markers that have been rejected from being included in the pair scan. show.selected.markers and show.rejected markers cannot be set to TRUE simultaneously.

show.selected.markers
A logical value. If select.markers.for.pairscan has been run, setting this value to TRUE places indicators above markers that have been selected for the pair scan.

Value
Nothing is returned from this function. It produces a plot of the effects.

See Also
singlescan, select.markers.for.pairscan

Examples

# plot all markers and both eigentraits
## Not run: plotSinglescan(obesity.cross)
# plot only results from chromosomes 1 through 4
## Not run: plotSinglescan(obesity.cross, chr = c(1:4))
plotSinglescan.heat  

Plot the results of singlescan as a heatmap

Description

This function plots the results obtained from the single-marker regression performed by singlescan. The effects (\(\beta\)) of each regression on each phenotype or eigentrait are plotted as heatmap. This method allows better comparison of effects between phenotypes than plotSinglescan.

Usage

plotSinglescan.heat(data.obj, standardized = TRUE,
show.marker.labels = FALSE, show.chr.boundaries = TRUE,
label.chr = TRUE, threshold.above = NULL,
color = "lightblue2", scale.fun = NULL)

Arguments

data.obj  The object in which all results are stored. See read.population.
standardized  A logical value. If TRUE, the absolute value of the regression t statistics (\(\beta/\sigma\)) are plotted. If FALSE, the raw regression coefficients (\(\beta\)) are plotted.
show.marker.labels  A logical value indicating whether marker names should be printed along the plot axes.
show.chr.boundaries  A logical value. If TRUE the boundaries of the chromosomes are marked with vertical lines.
label.chr  A logical value. If TRUE, the chromosomes are labeled with their numbers
threshold.above  A numeric value. Effects above this value are colored white. This is useful in preventing large effects from swamping out differences between smaller effects.
color  The color to be used in the raster image.
scale.fun  The name of function, such as sqrt or log10, that will scale the values in the matrix to increase contrast in the image.

Value

Nothing is returned from this function. It produces a heatmap of the effects

See Also

singlescan, plotSinglescan
plotSVD

Plot the results of the singular value decomposition of the phenotype matrix

Description

This function plots the results of the singular value decomposition (SVD) of the phenotype matrix. This plot is useful for selecting eigentraits for further analysis.

Usage

plotSVD(data.obj, orientation = c("vertical", "horizontal"))

Arguments

data.obj The object in which all results are stored. See read.population.
orientation A character string ("vertical", or "horizontal") indicating whether the plot should be vertically or horizontally oriented.

Value

No values are returned

See Also

gget.eigentraits

plotVariantInfluences

Plot variant-to-variant influences

Description

This function plots the reparameterized influences of variants on each other. The epistatic interactions from the pairwise scan are reparameterized to the terms $m_{12}$ and $m_{21}$, where the subscripts indicate the source and target variants respectively. These terms are interpreted as the effect that the source variant exerts on the target variant when both are present. Negative influences represent suppression while positive influences represent enhancement.

Usage

plotVariantInfluences(data.obj, p.or.q = 0.05, min.std.effect = 0, plot.all.vals = FALSE, all.markers = FALSE, standardize = TRUE, not.tested.col = "lightgray", show.marker.labels = FALSE, show.chr = TRUE, label.chr = TRUE, scale.effects = c("log10", "sqrt", "none"), pheno.width = 11, covar.width = 11, covar.labels = NULL, phenotype.labels = NULL)
Arguments

data.obj  The object in which all results are stored. See read.population.
p.or.q    A threshold indicating the maximum adjusted p value considered significant. If an fdr method has been used to correct for multiple testing, this value specifies the maximum q value considered significant. Only marker pairs with p or q values below this threshold will be plotted.
min.std.effect  A numerical threshold indicating the absolute value of the minimum standard effect size to be shown in the plot. The default value of 0 performs no thresholding.
plot.all.vals   A logical value indicating whether all values should be plotted regardless of significance. If TRUE, the significant values are highlighted in bright colors. If FALSE, non-significant values are not plotted.
all.markers       A logical value. If TRUE all markers are plotted. If FALSE only markers tested in the pair scan are plotted.
standardize       A logical value. If TRUE the values in each entry of the plotted matrix are the standardized effect sizes $\beta/\sigma$. If FALSE, the raw $\beta$ values are plotted.
not.tested.col   A color name used to mark variant pairs that were not tested due to linkage. The color distinguishes these pairs from those that were tested, but do not have significant interactions. The default color is light "gray". This can be changed to "white" or FALSE if no marking is desired.
show.marker.labels  A logical value indicating whether the marker labels should be printed along the plot axes.
show.chr         A logical value. If TRUE, chromosomes are indicated by alternating gray and white blocks along plot axes.
label.chr        A logical value. If TRUE and if show.chr is TRUE, chromosome numbers are printed inside each gray and white block.
scale.effects     A string indicating a scaling function by which the effects should be scaled. This is useful in increasing contrast between effects with large variance.
pheno.width      A numeric value indicating the width of the phenotypes relative to the width of each cell in the interaction matrix. Each cell in the interaction matrix has a width of 1, so a pheno.width of 10 makes the phenotypes 10 times wider for ease of viewing.
covar.width      A numeric value indicating the width of the covariates relative to the width of each cell in the interaction matrix. Each cell in the interaction matrix has a width of 1, so a covar.width of 10 makes the covariates 10 times wider for ease of viewing.
covar.labels     An optional vector of strings specifying a label for each covariate. If this argument is NULL, the covariate names from the data object will be used.
phenotype.labels An optional vector of strings specifying a label for each phenotype. If this argument is NULL, the phenotype names from the data object will be used.
Value

This function invisibly returns the plotted influence matrix.

See Also

pairscan

---

**read.population**

Read in and format data for analysis by cape

Description

This function reads in data for cape analysis and formats it into an object used by other functions in cape. A single comma-separated file containing both phenotype and genotype data is required. Chromosome and marker locations are required for each marker, and markers are assumed to be in order.

Usage

```r
read.population(filename = NULL, pheno.col = NULL, 
geno.col = NULL, delim = ",", na.strings = "-", 
check.chr.order = TRUE)
```

Arguments

- `filename`: An optional character string with path name specifying the file to be read in. Omission of this argument will prompt a dialog box for selecting a file.
- `pheno.col`: An optional numeric vector specifying which columns the phenotypes of interest are in. If omitted, all phenotypes are read in.
- `geno.col`: An optional numeric vector specifying which columns the genotypes of interest are in. If omitted, all genotypes are read in.
- `delim`: A character string indicating the delimiter in the data file. The default indicates a comma-separated file (",").
- `na.strings`: The symbol used to denote missing data in the file.
- `check.chr.order`: A logical value indicating whether the order of the chromosomes should be checked. In general, chromosomes should be entered in increasing numerical value. CAPE does not sort chromosomes, and they will be plotted in the order in which they are entered. If the chromosomes have non-numeric and non-X or Y names, and cannot be checked appropriately, or an alternate order is desired, set check.chr.order to FALSE.
Details

All phenotype and genotype data must be contained in a single comma-separated file. The phenotypes should be listed in columns at the beginning of the file, followed by the genotype data. Each row of the file corresponds to one individual. The file must contain the following attributes:

• header: A header labeling each column is required
• chromosomes: The second line of the file must contain the chromosome on which each marker is found. This line should begin with empty spaces in the phenotype columns followed by a chromosome label for each marker.
• marker location: The third line of the file must contain the chromosomal locations of the markers. Like the line of chromosome labels, this line should begin with empty spaces in the phenotype columns followed by a chromosomal position for each marker.
• phenotypes: The phenotypes must be listed in the first columns of the file. All phenotypes are required to be numeric. Phenotypes that are not numeric must be coded numerically. For example sex can be coded as [0,1]. Missing values are indicated with the symbol specified by na.strings. The default symbol for na.strings is '--'
• genotypes: Genotypes may be coded in one of three different formats: (1) As letters, for example A,H,B, indicating homozygous for allele 1, heterozygous, and homozygous for allele 2 respectively. "H" must be used for heterozygotes, but the other genotypes may be coded with any other letters. (2) As the numbers 0,1,2 indicating homozygous for allele 1, heterozygous, and homozygous for allele 2 respectively. (3) As continuous probabilities of the presence of the reference allele. An individual homozygous for allele 1 would be coded as 0, a heterozygous individual as 0.5, and an individual homozygous for allele 2 as 1. The continuous probabilities allow for uncertainty in genotyping that is not automatically available in the A,H,B or 0,1,2 encodings.

Value

The file is converted to a list object that is used as the main argument in most functions. This object is always referred to as data.obj. All data and analysis results will eventually be stored in this object. Upon creation the data.obj contains four elements:

pheno A matrix containing the phenotype data for the population. Each phenotype is stored in a column, and individuals are stored in rows.

geno A matrix containing the genotype data for the population. Each genotype is stored in a column, and individuals are stored in rows. Regardless of original format, the genotypes are converted to probabilities for in the data object. Genotypes originally coded as A,H,B for example, will be encoded as 0,0.5,1 respectively.

chromosome A vector containing the chromosome on which each marker is found.

marker.location A vector containing the chromosomal position of each marker.
select.by.chr

Subset a cross object to include only specified chromosomes.

Description

This function subsets a data object to include only specified chromosomes.

Usage

select.by.chr(data.obj, chr, include.covariates = TRUE)

Arguments

data.obj The object in which all results are stored. See read.population.

chr A vector of desired chromosomes

include.covariates Covariates are stored in the genotype matrix. To include these in the final subset, set include.covariates to TRUE. If include.covariates is FALSE, covariates will not automatically be included in the final genotype matrix.

Value

This function returns the data object with the genotype matrix pared down to only the desired chromosomes.

See Also

select.by.ind, create.covar

Examples

data(obesity.cross)
obesity.cross <- create.covar(obesity.cross, "mom")
# Subset the cross to only include chromosome 6 and the covariates
obesity.cross <- select.by.chr(obesity.cross, c(6))
str(obesity.cross)
select.by.ind

Subset a cross object to include specific individuals

Description

This function subsets a cross to include individuals based on either phenotypic or genotypic values. For example, this function can subset a cross to include all individuals with a phenotype value greater than x or a genotype value equal to y.

Usage

select.by.ind(data.obj, geno.or.pheno = pheno, expr)

Arguments

data.obj The object in which all results are stored. See read.population.
geno.or.pheno A character value, either "geno", or "pheno" to specify which matrix should be used to subset individuals.
expr An quoted expression used to subset individuals.

Value

The cross object is returned including only the individuals meeting the criteria in the provided expression.

See Also

select.by.chr

Examples

data(obesity.cross)
hist(obesity.cross$pheno[,"insulin"], main = "original insulin distribution", xlab = "insulin (ng/mL)", xlim = c(0, 30))

obesity.cross <- select.by.ind(obesity.cross, "pheno", "insulin < 25")
hist(obesity.cross$pheno[,"insulin"], main = "subset insulin distribution", xlab = "insulin (ng/mL)", xlim = c(0, 30))
select.eigentraits  Select a subset of the eigentraits for further analysis

Description

This function selects the specified eigentraits for further analysis. After the singular value decomposition (SVD) of multiple traits by `get.eigentraits`, a subset of the eigentraits, for example the first two, may carry useful information, while others may be dominated by noise. In this case, this function can be used to select the first two eigentraits for use in the analysis.

Usage

```r
select.eigentraits(data.obj, traits.which = c(1, 2))
```

Arguments

- `data.obj`: The object in which all results are stored. See `read.population`.
- `traits.which`: A numeric vector indicating which eigentraits should be retained for further analysis.

Details

Before the selection of eigentraits, the eigentraits should be examined using `plotSVD`.

Value

This function returns the data object with only the selected eigentraits.

References


See Also

`get.eigentraits`, `plotSVD`
select.markers.for.pairscan

A required step that filters variable and non-redundant markers for the pairscan

Description

This function selects markers for the pair scan. If the markers are to be thresholded by a significance cutoff, this function filters them. It then checks to make sure all pairs of markers have had at least one recombination between them, and that all markers are variable across individuals. Mono-allelic markers are removed. If any pair of markers carry identical genotype information, the first marker of the pair is discarded.

Usage

select.markers.for.pairscan(data.obj, 
use.pairs.threshold = FALSE, 
pairscan.thresh = NULL, specific.markers = NULL, 
num.markers = NULL, start.thresh = 4, 
tolerance = 10, verbose = TRUE)

Arguments

data.obj          The object in which all results are stored. See read.population.
use.pairs.threshold
                  A logical value. If TRUE only markers that fall above the significance threshold alpha.for.pairs (see singlescan). If FALSE, all markers are used.
pairscan.thresh   A numerical value indicating a standardized effect size below which markers are rejected for the pair scan.
specific.markers  An optional vector of column numbers or names specifying specific markers to include in the pairscan.
num.markers       An optional number. If specified, the algorithm attempts to find a standardized effect threshold that will give the number of markers specified. This can take a long time, since linear non-independence must be calculated for each thresholded set of markers.
start.thresh      If num.markers is specified, start.thresh specifies the standardized effect size at which the algorithm begins its search for the appropriate threshold. This should be set as close as possible to the estimated final effect size to minimize the amount of time spent looking for a threshold.
tolerance         If num.markers is specified, tolerance determines how close the algorithm will get to the number of desired markers. If 100 markers are desired, and tolerance is set to 10, the algorithm will stop if it finds a threshold giving between 90 and 110 markers.
verbose A logical value indicating whether the progress of the threshold finding algorithm should be reported.

Details

The markers that were selected for independence can be visualized with the function plotSinglescan with either show.selected.markers or show.rejected.markers set to TRUE.

Value

This function returns the data object with two new elements:

- geno.for.pairscan
  A matrix of the markers to be used in the pair scan. This matrix has the same structure as the original genotype matrix (see read.population), but contains only the filtered markers.

- covar.for.pairscan
  A matrix holding flags for whether each marker should be treated as a covariate. The matrix has one row for each of the filtered markers and one column for each eigentrait being analyzed. The entries of the matrix contain a 1 if the marker is to be used as a covariate for the pair scan and 0 otherwise.

See Also

singlescan, get.linearly.independent, plotSinglescan

select.pheno Select phenotypes for analysis

Description

This function is used to pare down a matrix of phenotypes to only those that will be included in the analysis.

Usage

select.pheno(data.obj, phenotypes)

Arguments

data.obj The object in which all results are stored. See read.population.
phenotypes A vector of either column numbers of the desired phenotypes, or character strings that uniquely identify the columns containing the desired phenotypes.

Value

This function returns the data object with a phenotype matrix that contains only the specified phenotypes.
Examples

```r
data(obesity.cross)
obesity.cross <- select.pheno(obesity.cross, c("glucose", "body_weight"))
```

Description

This function takes in a vector of markers and flags them for use as covariates in the pairwise scan if `is.covar` is TRUE. If `is.covar` is FALSE, the specified markers flagged such that they are not used as covariates in the pairwise scan. This function can be useful if multiple markers in an LD block exceed the pairscan threshold and cannot be selected automatically. Using all markers in one block as covariates may not be desired as it will prevent any of the markers in the block from having significant direct influences on any of the phenotypes.

Usage

```r
set.covar(data.obj, markers, pheno = NULL, is.covar = TRUE, plot.covar = TRUE)
```

Arguments

- **data.obj**: The object in which all results are stored. See `read.population`.
- **markers**: A vector specifying which markers are being specified as covariates or not covariates. The vector can specify the phenotypes either numerically or with characters, but the specification must be consistent for all entries.
- **pheno**: A vector specifying which phenotype(s) should be considered. Markers are set to be used as covariates for each phenotype independently. The vector can specify the phenotypes either numerically or with characters, but the specification must be consistent for all entries. If set to NULL, the marker is assigned to be a covariate for all phenotypes.
- **is.covar**: A logical value. If TRUE, the specified markers are flagged as covariates. If FALSE, the specified markers are flagged as non-covariates.
- **plot.covar**: A logical value. If TRUE, the results of the covariate specification are plotted. This plot is identical to the plot generated by `plotsinglescan`. Markers that are specified as covariates are colored red.

Value

This function operates on the element in data.obj called `covar.flags`. This element is a table with one row for each marker and one column for each trait being analyzed. The entry for each marker contains a 1 if it is to be used as a covariate and a 0 otherwise.

References

set.pairscan.thresh

**Description**

This function sets the threshold used to filter genetic markers for the pair scan. The threshold is a numeric t statistic \( \frac{\beta}{\sigma} \). If use of the threshold is indicated in `select.markers.for.pairscan`, only markers with t statistics exceeding the threshold will be selected for the pairscan. Thresholding is useful when the number of genotyped markers is large and an exhaustive pairwise scan is impractical.

**Usage**

```r
set.pairscan.thresh(data.obj, pairscan.thresh)
```

**Arguments**

- `data.obj` The object in which all results are stored. See `read.population`.
- `pairscan.thresh` A numeric value indicating the t statistic \( \frac{\beta}{\sigma} \) to be used as the threshold.

**Value**

This function returns the data object containing the specified pairscan threshold.

**References**


**See Also**

`select.markers.for.pairscan, pairscan`
singlescan

Run the single-variant regression for all phenotypes

Description

This function runs the single-variant regression for either raw phenotypes or eigentraits. It also performs permutation testing and adds two significance thresholds to the data object. One threshold optionally determines the significance cutoff for pairwise testing, and the other optionally determines the significance cutoff for which markers will be used as covariates in the pairwise scan.

Usage

singlescan(data.obj, n.perm = NULL, covar = NULL, scan.what = c("eigentraits", "raw.traits"), alpha = c(0.01, 0.05), verbose = FALSE)

Arguments

data.obj The object in which all results are stored. See read.population.
n.perm The number of permutations to be performed.
covar A vector of marker names to be used as covariates. See create.covar.
scan.what A character string indicating uniquely whether raw traits or eigentraits should be tested.
alpha A vector of alpha values for which significant standardized effect values will be calculated.
verbose A logical value indicating whether the progress of the scan should be printed to the screen.

Value

The data object is returned with two additional elements:
	singlescan.results A list in which each element corresponds to one trait being scanned. Each element contains a table with the results from the single-marker scan. The table includes, in columns, the effect size marker, the standard error of the effect size, the t statistic from the regression, and the p value. There is one row for each marker.

covar.flags A table indicating which variant, in rows, should be used as a covariate in the scan of each phenotype, in columns. A variant is designated as a covariate with a 1.

References

See Also

select.markers.for.pairscan, set.covar, get.covar

writePopulation
Write out a cape data object to .csv format.

Description

This function takes in a data object and writes it to a .csv file. This file can be read in by read.population.

Usage

writePopulation(data.obj, filename,
    geno.to.convert = NULL, conversion = NULL)

Arguments

data.obj The object in which all results are stored. See read.population.
filename A string specifying the name of the file to be written.
genotocovert A vector of genotypes in the data object that are to be converted to another form. For example, to convert genotypes coded as (0, 0.5, 1), to ("A", "H", "B"), set geno.to.covartocovar to c(0, 0.5, 1) and conversion to c("A", "H", "B").
conversion A vector the same length as geno.to.covartocovar. The values in this vector will be written in place of the values in geno.to.convert in the final file.

Value

This function writes a data object to a .csv file that can be read by read.population. Nothing is returned.

See Also

read.population
writeVariantInfluences

Write the final results to a file

Description

This function takes in the final data object and writes the variant influences that are at or below the specified significance level to a file in the current working directory.

Usage

```r
writeVariantInfluences(data.obj, p.or.q = 0.05,
filename = "Variant.Influences.csv", delim = ",
mark.covar = FALSE, write.file = TRUE)
```

Arguments

- `data.obj`: The object in which all results are stored. See `read.population`.
- `p.or.q`: A threshold indicating the maximum adjusted p value considered significant. If an fdr method has been used to correct for multiple testing, this value specifies the maximum q value considered significant. Only marker pairs with p or q values below this threshold will be plotted.
- `filename`: A character vector specifying the name of the file.
- `delim`: A character string indicating the delimiter in the data file. The default indicates a comma-separated file (",").
- `mark.covar`: A logical value. If TRUE, an asterisk is appended the names of markers used as covariates in the pair scan.
- `write.file`: A logical value indicating whether the table should be written to a file or simply returned.

Value

This function writes a table of direct influences to a file. It also returns the table invisibly, i.e. if the output of the function is assigned to an object, the object will contain the table of influences. Otherwise, nothing is returned.

Examples

```r
# here the table is written to a file, but nothing is returned.
## Not run: writeVariantInfluences(obesity.cross)

# here the table is written to a file, and returned to the
# object sig.table
## Not run:
sig.table <- writeVariantInfluences(obesity.cross)
print(sig.table)
## End(Not run)
```
Index

*Topic IO
  read.population, 22
  writePopulation, 32
  writeVariantInfluences, 33
*Topic algebra
  get.eigentraits, 8
*Topic arith
  direct.influence, 5
  get.covar, 8
  norm.pheno, 11
  set.covar, 29
*Topic datasets
  obesity.cross, 12
*Topic hplot
  plotCollapsedVarInf, 14
  plotNetwork, 15
  plotPairscan, 16
  plotSingleScan, 17
  plotSingleScan.heat, 19
  plotSVD, 20
  plotVariantInfluences, 20
*Topic manip
  create.covar, 4
  delete.pheno, 5
  get.network, 9
  norm.pheno, 11
  select.by.chr, 24
  select.by.ind, 25
  select.eigentraits, 26
  select.markers.for.pairscan, 27
  select.pheno, 28
  set.pairscan.thresh, 30
*Topic math
  error.prop, 7
*Topic models
  calc.p, 3
*Topic package
  cape-package, 2
*Topic regression
  pairscan, 12
  singlescan, 31
*Topic symbolmath
  error.prop, 7
calc.p, 3, 7
cape, 22
cape (cape-package), 2
cape-package, 2
create.covar, 4, 24, 30, 31
delete.pheno, 5
direct.influence, 5
error.prop, 3, 7
get.covar, 8, 30, 32
get.eigentraits, 2, 5, 8, 11, 20, 26
get.linearly.independent, 28
get.network, 9, 14–16
norm.pheno, 9, 11
obesity.cross, 12
pairscan, 3, 5, 8, 12, 15–17, 22, 30
plotCollapsedVarInf, 9, 10, 14
plotNetwork, 9, 10, 15
plotPairscan, 14, 16
plotSingleScan, 17, 19, 28–30
plotSingleScan.heat, 19
plotSVD, 9, 20, 26
plotVariantInfluences, 14–16, 20
read.population, 2–11, 13, 15–17, 19–21, 22, 24–33
select.by.chr, 24, 25
select.by.ind, 24, 25
select.eigentraits, 26
select.markers.for.pairscan, 8, 14, 18, 27, 30, 32

34
INDEX

select.pheno, 28
set.covar, 8, 29, 32
set.pairscan.thresh, 30
singlescan, 3, 8, 18, 19, 27, 28, 31
writePopulation, 32
writeVariantInfluences, 33